Math 142: Modeling **Name:**____________________________

Homework # 3 **Due: Friday Oct 20**

Directions Complete the exercises. Your solutions to the exercises should be submitted to Gradescope before the indicated due date above. Please follow rules regarding Gradescope submission as described in the syllabus.

References Except for the help of the instructor or TAs and the class textbooks and notes, if you use any resources, for example, a book, a website, or you discussed with your friends, please acknowledge them in this References section.

- I discussed Problem ?? with STUDENT A, STUDENT B, ...
- I used BOOK/WEBSITE to help me do Problem ??.

Exercises

1. Calculations using Leslie matrices
   (a) Consider the Leslie matrix and initial conditions:

$$L = \begin{pmatrix} 0.5 & 2 \\ 1 & 1.5 \end{pmatrix} \quad , \quad \boldsymbol{N}_0 = \begin{pmatrix} 20 \\ 10 \end{pmatrix}.$$

   (i) Derive an explicit formula for $\boldsymbol{N}_k$.
   $\boldsymbol{N}_k = L^k \boldsymbol{N}_0 = \begin{pmatrix} 0.5 & 2 \\ 1 & 1.5 \end{pmatrix}^k \begin{pmatrix} 20 \\ 10 \end{pmatrix} = \frac{20}{3}(-0.5)^k \begin{pmatrix} 1 \\ -0.5 \end{pmatrix} + \frac{40}{3}(2.5)^k \begin{pmatrix} 1 \\ 1 \end{pmatrix}$

   (ii) We have used whether the largest eigenvalue of L is > 1 to argue whether or not populations will grow exponentially. Explain why there are no realistic initial conditions for this model for which the population will not grow exponentially. (Hint: For initial conditions to be realistic, every subpopulation must have a non-negative size).
   Let $f(x) := \det(L - xI) = (0.5 - x)(1.5 - x) - 2 \Rightarrow \lambda_0 = -0.5, \lambda_1 = 2.5$ are eigenvalues of matrix L with corresponding eigenvectors $x_0 = \begin{pmatrix} 1 \\ -0.5 \end{pmatrix}, x_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$. Suppose we have initial condition $\boldsymbol{N}_0 = \begin{pmatrix} n_0 \\ n_1 \end{pmatrix}$ where $n_0, n_1 \geq 0$. It follows there exist a_0, a_1 s.t $\boldsymbol{N}_0 = a_0 x_1 + a_1 x_1$ because x_0, x_1 are linearly independent. It follows $a_0 + a_1 = n_0$ and $-0.5a_0 + a_1 = n_1$. Suppose $a_1 < 0 \Leftrightarrow a_0 > n_0$. Because $n_0 \geq 0 \Rightarrow a_0 > 0$. Thus, $-0.5a_0 + a_1 < 0 \Rightarrow n_1 < 0$ which is impossible under our assumption of realistic conditions. Hence we have a dominating eigenvalue of 2.5 with $a_1 > 0$, so $\lim_{k\to\infty} N_k = a_1 \lambda_1^k x_1$ which each component will exponentially grow toward infinity.

   (b) Consider a Leslie matrix representing transitions among three age groups: 0,1 and 2 :

$$L = \begin{pmatrix} 0 & 3 & 1.5 \\ 1 & 0 & 0 \\ 0 & 1 & 0.5 \end{pmatrix}.$$

(i) Using Python (or fav software) to help you, if necessary, derive an explicit formula for $\boldsymbol{N}_k$ if $\boldsymbol{N}_0 = \begin{pmatrix} 30 \\ 10 \\ 10 \end{pmatrix}$.

$$\boldsymbol{N}_k = \begin{pmatrix} -0.86 & 0.8 & 0 \\ -0.43 & -0.53 & -0.45 \\ -0.29 & 0.27 & 0.89 \end{pmatrix} \begin{pmatrix} 2 & 0 & 0 \\ 0 & -1.5 & 0 \\ 0 & 0 & 0 \end{pmatrix}^k \begin{pmatrix} 0 & -1 & -1 \\ 1 & -1 & -1 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 30 \\ 10 \\ 10 \end{pmatrix}$$

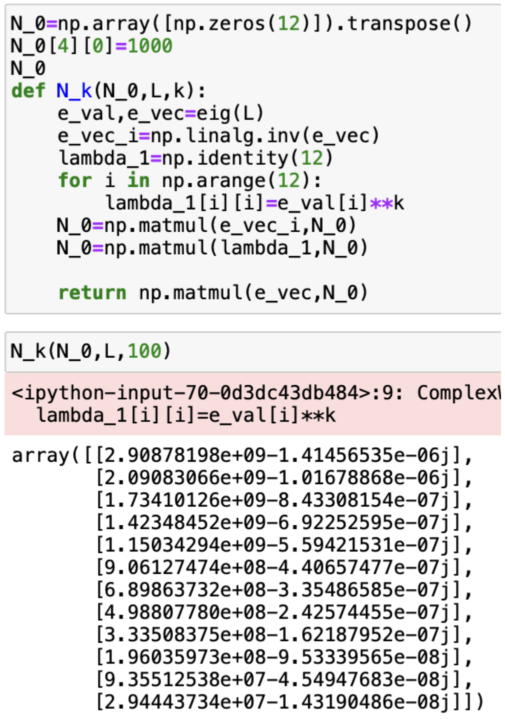
(ii) For general initial conditions find the proportion of organisms in the age 0 category when the population converges to its stable age distribution.
$0.\overline{54}$

2. The following table lists birth and **survivor** rates for the female population of a certain species of domestic sheep in New Zealand, where sheep farming is a major segment of the economy (Data from G. Caughley, "Parameters for Seasonally Breeding Populations"). Sheep give birth only once a year. In the species under consideration, sheep seldom, if ever, live longer than 12 years.

Age (years)	Birth Rate	Survival Rate
0-1	0.000	0.845
1-2	0.045	0.975
2-3	0.391	0.965
3-4	0.472	0.950
4-5	0.484	0.926
5-6	0.546	0.895
6-7	0.543	0.850
7-8	0.502	0.786
8-9	0.468	0.691
9-10	0.459	0.561
10-11	0.433	0.370
11-12	0.421	0.000

(a) Use Python (or fav software) to calculate the largest eigenvalue of your matrix. Will the population be growing or decaying? Why?
The largest eigenvalue is $\lambda_0 = 1.1755714232604326$. Since the dominating eigenvalue is greater than 1, we should expect the population to grow.

(b) If there are 1000 female sheep in the age rage of 4-5 years initially (for our simulation), how many sheep in each age range will there be in 100 years?



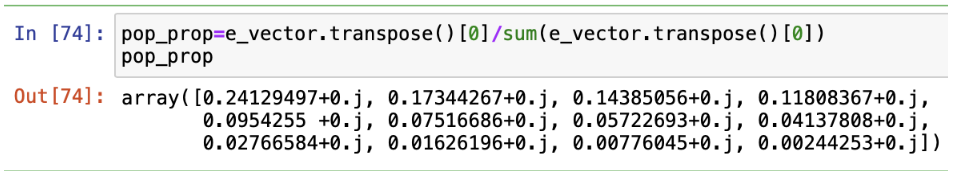
```
N_0=np.array([np.zeros(12)]).transpose()
N_0[4][0]=1000
N_0
def N_k(N_0,L,k):
    e_val,e_vec=eig(L)
    e_vec_i=np.linalg.inv(e_vec)
    lambda_1=np.identity(12)
    for i in np.arange(12):
        lambda_1[i][i]=e_val[i]**k
    N_0=np.matmul(e_vec_i,N_0)
    N_0=np.matmul(lambda_1,N_0)

    return np.matmul(e_vec,N_0)
```

```
N_k(N_0,L,100)
```

```
<ipython-input-70-0d3dc43db484>:9: Complex
  lambda_1[i][i]=e_val[i]**k
```

```
array([[2.90878198e+09-1.41456535e-06j],
       [2.09083066e+09-1.01678868e-06j],
       [1.73410126e+09-8.43308154e-07j],
       [1.42348452e+09-6.92252595e-07j],
       [1.15034294e+09-5.59421531e-07j],
       [9.06127474e+08-4.40657477e-07j],
       [6.89863732e+08-3.35486585e-07j],
       [4.98807780e+08-2.42574455e-07j],
       [3.33508375e+08-1.62187952e-07j],
       [1.96035973e+08-9.53339565e-08j],
       [9.35512538e+07-4.54947683e-08j],
       [2.94443734e+07-1.43190486e-08j]])
```

(c) What is the stable age distribution?



```
In [74]: pop_prop=e_vector.transpose()[0]/sum(e_vector.transpose()[0])
         pop_prop

Out[74]: array([0.24129497+0.j, 0.17344267+0.j, 0.14385056+0.j, 0.11808367+0.j,
                0.0954255 +0.j, 0.07516686+0.j, 0.05722693+0.j, 0.04137808+0.j,
                0.02766584+0.j, 0.01626196+0.j, 0.00776045+0.j, 0.00244253+0.j])
```

(d) New Zealand sheep farmers cannot live entirely on their income from wool, especially if they have to keep feeding ever more sheep. A desirable goal for management of a sheep herd (or any renewable resource) is to find a stable configuration from which one can harvest the *growth* at regular intervals, thereby producing income and returning the population to its previous configuration. A sustainable harvesting policy is a plan for harvesting on a regular schedule in such a way that the harvest is always the same and the state of the population after harvesting is always the same.

Let h_i be the fraction of the i-th age group that will be harvested at the end of each growth period and H be the diagonal matrix whose entries are the h_i's. For the previous parts, $N^{(k+1)} = LN^{(k)}$ but now if we consider the harvest after the growth, $N^{(k+1)} = LN^{(k)} - HLN^{(k)} = (I - H)LN^{(k)}$.

(i) We want a stable population, i.e. not changing at all. What eigenvalue corresponds to a stable population?
An eigenvalue less than or equal to 1 1 corresponds to a stable population

(ii) What conditions are on h_i's to have stable population?
Choose H s.t $\det((I - H)L - I) = 0$, so $(I - H)L$ has an eigenvalue of 1.

(e) A uniform harvesting policy is one in which the same fraction h is harvested from each age group. Explain why this means $h = 1 - \frac{1}{\lambda_0}$ (λ_0 is the largest eigenvalue). In the long term, the population will grow according to the dominating eigenvalue-eigenvector pair, so we want $\lambda_0(\vec{v_0}) - \lambda_0 H(\vec{v_0}) = \vec{v_0}$

3. Supppose that

$$L = \begin{bmatrix} 2 & 4 \\ 0.3 & 0 \end{bmatrix}$$

is the Leslie matrix for a population with two age classes. It's convenient to always use the computer to compute things for us, but we don't always have computers. Without using the computer or technology, compute the following

   (a) Determine both eigenvalues. Let $f(x) := \det(L - xI) = (2-x)(0-x) - 1.2 \Rightarrow \lambda_0 = 1 - \sqrt{2.2}, \lambda_1 = 1 + \sqrt{2.2}$

   (b) Find the stable age distribution. $x_1 = \begin{pmatrix} 4 \\ \sqrt{2.2} - 1 \end{pmatrix}$ $p = \begin{pmatrix} \frac{4}{3+\sqrt{2.2}} \\ \frac{\sqrt{2.2}-1}{3+\sqrt{2.2}} \end{pmatrix}$

4. In this question we are going to create a Leslie matrix model for the US population. We will have to collect data of birth and death rates from the National Center for Health Statistics.

   You can download the most recent (2017) data on mortality rates from: `https://ftp.cdc.gov/pub/Health_Statistics/NCHS/Publications/NVSR/68_07/`

   The different tables break the data down by race and by sex-at-birth. Table01 aggregates the data over the entire population. This is the table that we will use. The probability of dying data are all in column B.

   For birth rates we will use the data from 2017 published in Table 2 (Page 13) of the report:

   `https://www.cdc.gov/nchs/data/nvsr/nvsr68/nvsr68_13-508.pdf`

   One wrinkle - birth rates are reported for women, but our population model includes all individuals (including non-women). Assume that a fraction 0.508 of individuals are women (also based on CDC data). So multiply birth rates by 0.508 to account for the fact that we must find the fraction of individuals in the age group who are women, and then find the fraction of births.

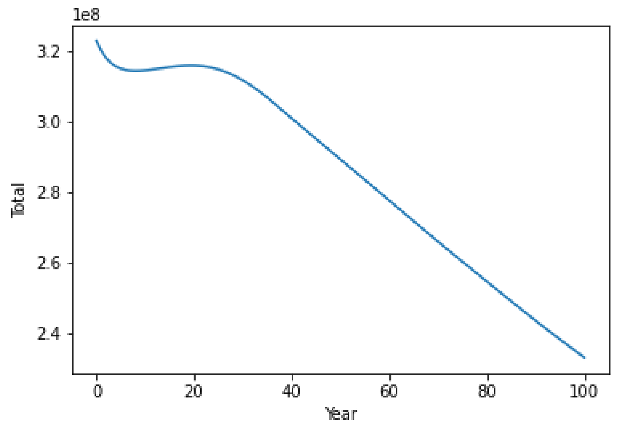
   Your first goal is to assemble a Leslie matrix using all of these data. For the birth rate data, which is aggregated over 5 years - e.g. 20-24 it is acceptable to assume that all ages in the group have the same birth rate - i.e. use the same birth rate for 20, 21, 22, 23 and 24 year olds. There is no need to write out or print the matrix - just use it to answer the questions.

   (a) Why might it be inaccurate to assume that the same fraction of individuals in each age group are women? In other words why might different age groups have different fractions of women?
   One reason is that women on average live longer than men, so we'd expect to see a higher percentage of women in older populations. Other external factors such as a war could affect the proportion of each gender for a given age group.

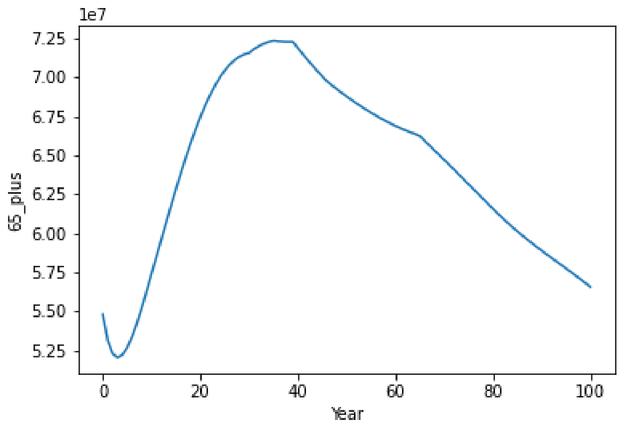
   (b) Use Python (or fab software) to calculate the largest eigenvalue of your matrix. Is it larger than 1, meaning that the population will increase over time?
   the largest eigenvalue is 0.9956059820629688 which is less than 1, so the population is decreasing.

   (c) Find for yourself the most recent data on the distribution of ages of individuals in the US (cite the website or source that you used). Using your Leslie matrix model, predict how the populations will grow or decay over the next 100 years. Demonstrate your answer by making the following plots:
   `https://www.kff.org/other/state-indicator/distribution-by-age/?dataView=1&currentTimeframe`
   `0&selectedRows=%7B%22wrapups%22:%7B%22united-states%22:%7B%7D%7D%7D&sortModel=%7B%`
   `22colId%22:%22Location%22,%22sort%22:%22asc%22%7D`

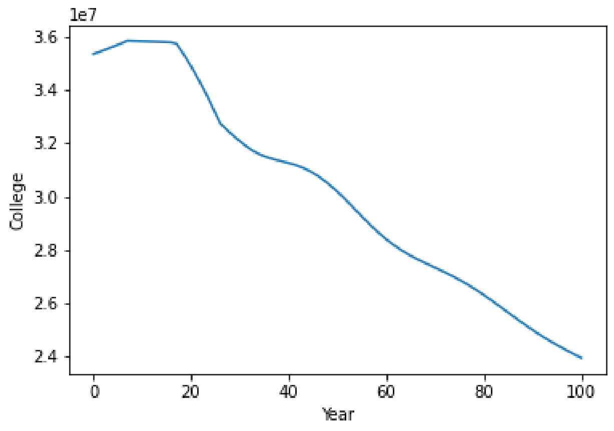
(i) The total population over time.



(ii) The number of people over the age of 65 as a fraction of the total population size.



(iii) The number of college age (i.e. 18-25) people as a function of the total population size.



(d) What important effect did we miss when trying to predict demographic change in the US? Immigration and emmigration.

5. Submit the code you used for any and all of the problems. (Print pdf the code) If we have any questions, we may ask you for the actual code file.
Used for data: `https://www.kff.org/other/state-indicator/distribution-by-age/?dataView=1&currentTimeframe=0&selectedRows=%7B%22wrapups%22:%7B%22united-states%22:%7B%7D%7D%7D&sortModel=%7B%22colId%22:%22Location%22,%22sort%22:%22asc%22%7D`

In [2]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

from numpy.linalg import eig
```

In [3]:
```
L=np.array([[0,3,1.5],
            [1,0,0],
            [0,1,0.5]])
```

In [98]:
```
e_val,e_vec=eig(L)
```

In [100…
```
np.round(e_vec,2)
```

Out[100…
```
array([[-0.86,  0.8 , -0.  ],
       [-0.43, -0.53, -0.45],
       [-0.29,  0.27,  0.89]])
```

In [102…
```
np.round(np.linalg.inv(e_vec))
```

Out[102…
```
array([[-0., -1., -1.],
       [ 1., -1., -1.],
       [-0.,  0.,  1.]])
```

In [101…
```
lambda_1=np.identity(3)
for i in np.arange(3):
    lambda_1[i][i]=e_val[i]
np.round(lambda_1,2)
```

Out[101…
```
array([[ 2. ,  0. ,  0. ],
       [ 0. , -1.5,  0. ],
       [ 0. ,  0. ,  0. ]])
```

In [81]:
```
N_0=np.array([[30,10,10]]).transpose()
```

In [96]:
```
def N_k(N_0,L,k):
    e_val,e_vec=eig(L)
    e_vec_i=np.linalg.inv(e_vec)
    lambda_1=np.identity(3)
    for i in np.arange(3):
        lambda_1[i][i]=e_val[i]**k
    N_0=np.matmul(e_vec_i,N_0)
    N_0=np.matmul(lambda_1,N_0)

    return np.matmul(e_vec,N_0)
```

In [97]:
```
N_k(N_0,L,1)
```

Out[97]:
```
array([[45.],
       [30.],
       [15.]])
```

In [87]:
```
e_vec.transpose()[0][0]/np.sum(e_vec.transpose()[0])
```

Out[87]:
```
0.5454545454545454
```

In [1]:
```python
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

from numpy.linalg import eig
```

In [78]:
```python
b=np.array([0.000,0.045,0.391,0.472,0.484,0.546,0.543,0.502,0.468,0.459,0.433,0.421])
S=np.array([0.845,0.975,0.965,0.950,0.926,0.895,0.850,0.786,0.691,0.561,0.370])
L=np.zeros((12,12))
L[0]=b
for i in np.arange(1,12):
    L[i][i-1]=S[i-1]
```

In [80]:
```python
L
```

Out[80]:
```
array([[0.   , 0.045, 0.391, 0.472, 0.484, 0.546, 0.543, 0.502, 0.468,
        0.459, 0.433, 0.421],
       [0.845, 0.   , 0.   , 0.   , 0.   , 0.   , 0.   , 0.   , 0.   ,
        0.   , 0.   , 0.   ],
       [0.   , 0.975, 0.   , 0.   , 0.   , 0.   , 0.   , 0.   , 0.   ,
        0.   , 0.   , 0.   ],
       [0.   , 0.   , 0.965, 0.   , 0.   , 0.   , 0.   , 0.   , 0.   ,
        0.   , 0.   , 0.   ],
       [0.   , 0.   , 0.   , 0.95 , 0.   , 0.   , 0.   , 0.   , 0.   ,
        0.   , 0.   , 0.   ],
       [0.   , 0.   , 0.   , 0.   , 0.926, 0.   , 0.   , 0.   , 0.   ,
        0.   , 0.   , 0.   ],
       [0.   , 0.   , 0.   , 0.   , 0.   , 0.895, 0.   , 0.   , 0.   ,
        0.   , 0.   , 0.   ],
       [0.   , 0.   , 0.   , 0.   , 0.   , 0.   , 0.85 , 0.   , 0.   ,
        0.   , 0.   , 0.   ],
       [0.   , 0.   , 0.   , 0.   , 0.   , 0.   , 0.   , 0.786, 0.   ,
        0.   , 0.   , 0.   ],
       [0.   , 0.   , 0.   , 0.   , 0.   , 0.   , 0.   , 0.   , 0.691,
        0.   , 0.   , 0.   ],
       [0.   , 0.   , 0.   , 0.   , 0.   , 0.   , 0.   , 0.   , 0.   ,
        0.561, 0.   , 0.   ],
       [0.   , 0.   , 0.   , 0.   , 0.   , 0.   , 0.   , 0.   , 0.   ,
        0.   , 0.37 , 0.   ]])
```

In [65]:
```python
e_value,e_vector=eig(L)
```

In [73]:
```python
e_value
```

Out[73]:
```
array([ 1.17557142+0.j        ,  0.56162493+0.54934695j,
        0.56162493-0.54934695j,  0.23526738+0.6945441j ,
        0.23526738-0.6945441j , -0.1301436 +0.71474201j,
       -0.1301436 -0.71474201j, -0.35689924+0.56111754j,
       -0.35689924-0.56111754j, -0.6585493 +0.j        ,
       -0.56836052+0.31311574j, -0.56836052-0.31311574j])
```

In [66]:
```python
e_value[0]
```

Out[66]:
```
(1.1755714232604326+0j)
```

In [70]:
```python
N_0=np.array([np.zeros(12)]).transpose()
N_0[4][0]=1000
N_0
def N_k(N_0,L,k):
    e_val,e_vec=eig(L)
    e_vec_i=np.linalg.inv(e_vec)
    lambda_1=np.identity(12)
    for i in np.arange(12):
        lambda_1[i][i]=e_val[i]**k
    N_0=np.matmul(e_vec_i,N_0)
    N_0=np.matmul(lambda_1,N_0)

    return np.matmul(e_vec,N_0)
```

In [71]:
```python
N_k(N_0,L,100)
```

```
<ipython-input-70-0d3dc43db484>:9: ComplexWarning: Casting complex values to real discards the imaginary part
  lambda_1[i][i]=e_val[i]**k
```

Out[71]:
```
array([[2.90878198e+09-1.41456535e-06j],
       [2.09083066e+09-1.01678868e-06j],
       [1.73410126e+09-8.43308154e-07j],
       [1.42348452e+09-6.92252595e-07j],
       [1.15034294e+09-5.59421531e-07j],
       [9.06127474e+08-4.40657477e-07j],
       [6.89863732e+08-3.35486585e-07j],
       [4.98807780e+08-2.42574455e-07j],
       [3.33508375e+08-1.62187952e-07j],
       [1.96035973e+08-9.53339565e-08j],
       [9.35512538e+07-4.54947683e-08j],
       [2.94443734e+07-1.43190486e-08j]])
```

In [62]:
```python
e_vector.transpose()[0]
```

Out[62]:
```
array([0.63635094+0.j, 0.45740866+0.j, 0.37936737+0.j, 0.3114141 +0.j,
       0.25165922+0.j, 0.19823248+0.j, 0.15092071+0.j, 0.10912361+0.j,
       0.07296125+0.j, 0.04288657+0.j, 0.0204661 +0.j, 0.00644151+0.j])
```

In [74]:
```python
pop_prop=e_vector.transpose()[0]/sum(e_vector.transpose()[0])
pop_prop
```

Out[74]:
```
array([0.24129497+0.j, 0.17344267+0.j, 0.14385056+0.j, 0.11808367+0.j,
       0.0954255 +0.j, 0.07516686+0.j, 0.05722693+0.j, 0.04137808+0.j,
       0.02766584+0.j, 0.01626196+0.j, 0.00776045+0.j, 0.00244253+0.j])
```

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

from numpy.linalg import eig
```

In [22]:
```
b=np.array([0.2,18.8,71.0,98.0,100.3,52.3,11.6,0.9])/1000*0.508
```

In [3]:
```
df=pd.read_excel("Table18.xlsx")
```

In [17]:
```
m=df.iloc[2:102,1].to_numpy()
```

In [88]:
```
L=np.zeros((100,100))
k=0
for i in np.arange(10,50,5):
    for j in np.arange(i,i+5):
        L[0][j]=b[k]
    k+=1
for l in np.arange(0,99):
    L[l+1][l]=1-m[l]
```

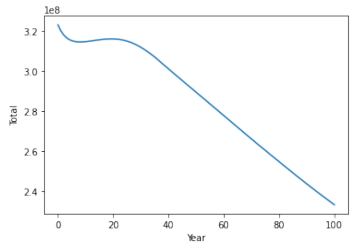
In [84]:
```
e_val,e_vec=eig(L)
```

In [121…
```
max(e_val)
```

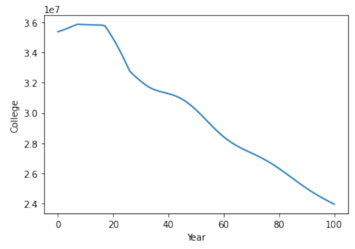
Out[121…
```
(0.9956059820629688+0j)
```

In [117…
```
N_0=np.array([np.zeros(100)])
for i in np.arange(19):
    N_0[0][i]=75887400/19
for i in np.arange(19,26):
    N_0[0][i]=27366000/7
for i in np.arange(26,35):
    N_0[0][i]=39656400/9
for i in np.arange(35,55):
    N_0[0][i]=83093900/20
for i in np.arange(55,65):
    N_0[0][i]=42413200/10
for i in np.arange(65,100):
    N_0[0][i]=54798900/35
N_0=N_0.transpose()
Pop=pd.DataFrame({"Year":[],"Total":[],"College":[],"65_plus":[]})
for i in np.arange(101):
    if i>0:
        N_0=np.matmul(L,N_0)

    x={"Year":i,"Total":np.sum(N_0),"College":np.sum(N_0[17:26]),"65_plus":np.sum(N_0[65:])}
    Pop=Pop.append(x,ignore_index=True)
```

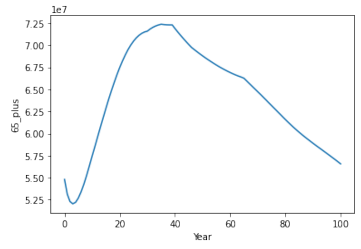
In [125…
```
sns.lineplot(data=Pop,x="Year",y="Total")
plt.savefig("Total vs Time")
```



In [126…
```
sns.lineplot(data=Pop,x="Year",y="College")
plt.savefig("College vs Time")
```



In [127…
```
sns.lineplot(data=Pop,x="Year",y="65_plus")
plt.savefig("65+ vs Time")
```



In [ ]: